

Making DBRMs More Readable (or Playing With Strings)

Ann Stephan, Texas State Comptroller, Austin, TX

I have attended several papers on working with character strings and usually thought "That's nice, but why you would ever want to do that?" I realize now that I have been taking it for granted that any data I needed to read would be nice, clean, and easily readable. I have generally been reading tax or tax-related data which had been processed and stored in VSAM or DB2. When I took my first look at a DBRM, it looked like a jumble.

A DBRM (database request module) is created by the DB2 precompiler and contains information about SQL statements. Each SQL select statement is captured, "DBRM" is placed at the beginning and some code is added which is meaningful to DB2 but not necessary for the programmers when debugging. The DBRMs are generated in 80 byte chunks - with no regard for word splits. There are also extra spaces thrown in before & after each parentheses, comma, or period. When the programmers (Natural, COBOL, or PowerBuilder) were trying to pinpoint a problem, they needed to be able to locate the SQL causing the problem, cut & paste it to make changes and test it. Since we can see only 72 columns at a time, looking at the 80 column DBRMS required paging left and right. Trying to read them was a very frustrating process, so I was asked to take a look & see if I could generate something a little more user-friendly than what they were trying to work with.

The first line of each DBRM contains the plan name, and then some special characters so only the first 20 characters are read in for the first line. The first line of each DBRM piece starts with the word DBRM - then more special characters - with the SQL starting in column 25. These lines are read in with the special characters omitted. For the subsequent lines in each chunk, the entire 80 characters are read. Each line is numbered and each DBRM numbered.

```
000010 DATA DBRM;          INFILE DBRMS;
000011     LENGTH TEXT $80;
000012     RETAIN LN 0 DN 0;
000013     LN+1;
000014     INPUT @ 1 CKDBRM $4.@;
000016     IF LN = 1 THEN DO;
000017         INPUT @1 TEXT $23. ;
000018         END;
000019     ELSE IF LN > 1 THEN DO;
000021         IF CKDBRM NE 'DBRM' THEN DO;
000022             INPUT @ 1 TEXT $80. ;
000023             END;
000024         ELSE IF CKDBRM = 'DBRM' THEN DO;
000025             INPUT @ 25 TEXTX $56.;
000026             DN+1;
000027             TEXT=CKDBRM||' '||TEXTX;
000028             END;
000029         END;
000030 RUN
```

"For fetch only" can appear anywhere in a line, and if it appears in a line which is not the last line of the group there can be one or more lines of junk following. A flag is set containing the line number of the line containing "for fetch only". The flag is put in a separate dataset.

```

000032 DATA DBRM(DROP=FLAG2) CKFLAG2(KEEP=DN FLAG2);          SET DBRM;
000033                                     BY DN;
000034     RETAIN CT 0;
000036     IF FIRST.DN THEN CT=0;
000037     CT+1;
000039     LEN=LENGTH(TEXT);
000040     DO I=1 TO LEN-14;
000041     FET = 'FOR FETCH ONLY';
000042     C2TEXT=SUBSTR(TEXT,I,14);
000043     IF C2TEXT=FET THEN DO;
000044     XTEXT=SUBSTR(TEXT,1,I+14);
000045     TEXT=XTEXT;
000046     LEN=I;
000047     IF NOT LAST.DN THEN DO;
000048     FLAG2=CT;
000049     OUTPUT CKFLAG2;
000050     END;
000051     END;
000052     END;
000053     OUTPUT DBRM;
000054 RUN;

```

The 2 datasets are then merged to put the flag with the rest of the lines, and any lines greater than the flagged line are deleted.

```

000055 DATA DBRM; MERGE DBRM (IN=D)
000056             CKFLAG2 (IN=F);
000057             BY DN;
000058     IF D=1;
000059     IF F=1 & CT > FLAG2 THEN DELETE;
000060     ELSE OUTPUT DBRM;
000061 RUN;

```

Next, each line is read in 3 character chunks (from position 1 to the length of the line) - looking for the special characters with the extra spaces. The I-counter is used to determine the position of the found character. When one is encountered, those 3 bytes are replaced by the character - without the surrounding spaces. I have had a problem with the spacing around the "("s, and I will continue to work on that, but the PowerBuilder guys are happy with the results so far.

```

000063 DATA COMP; SET DBRM;
000064     LENGTH TEXT1 $80;
000065
000066     RETAIN LEN I; LENGTH CK1 CK2 CK3 CK4 CKTEXT FLAG $3 TEXT $80;
000067     LEN=LENGTH(TEXT);
000068
000069     CK1 = ' . ';
000070     CK2 = ' , ';
000071     CK3 = ' ( ';
000072     CK4 = ' ) ';
000073
000074     DO I=1 TO LEN;
000075
000076     IF I >=LEN-1 THEN CKTEXT=SUBSTR(TEXT,I);
000077     ELSE CKTEXT=SUBSTR(TEXT,I,3);
000078     IF CKTEXT = CK1 OR CKTEXT=CK2 OR CKTEXT=CK3 OR CKTEXT=CK4

```

```

000079 THEN DO;
000080 IF CKTEXT = CK1 THEN DO;
000081     FLAG='CK1';
000082     SUBSTR(TEXT,I,2)='.';
000083     END;
000084 ELSE IF CKTEXT = CK2 THEN DO;
000085     FLAG='CK2';
000086     SUBSTR(TEXT,I,2)=',';
000087     END;
000088 ELSE IF CKTEXT = CK3 THEN DO;
000089     FLAG='CK3';
000090     SUBSTR(TEXT,I,2)='(';
000091     END;
000092 ELSE IF CKTEXT = CK4 THEN DO;
000093     FLAG='CK4';
000094     SUBSTR(TEXT,I,2)=')';
000095     END;

```

The line length is reset, and the line is then broken into two pieces: the first is from 1 to I, the second is from I+1 to the line-length. If I is greater than or equal to the length-4 the second piece is set to spaces. Then the two pieces are put back together. If the character is a “,” or “)” a space is put between the two strings. If the character is a “.” or “(” no space is added.

```

000096     LEN=LENGTH(TEXT);
000097     IF I < LEN-4 THEN DO;
000098         TEXT2=SUBSTR(TEXT,I+1);
000099         TEXT1=SUBSTR(TEXT,1,I);
000100         END;
000101     ELSE IF I >=LEN-4 THEN DO;
000102         TEXT2=' ';
000103         TEXT1=TEXT;
000104         END;
000105     IF CKTEXT=CK2 OR CKTEXT=CK4 THEN
000106         TEXT=TRIM(TEXT1)||' '||LEFT(TEXT2);
000107     ELSE IF CKTEXT=CK1 OR CKTEXT=CK3 THEN
000108         TEXT=TRIM(TEXT1)||LEFT(TEXT2);
000109     END;
000110 END;
000111 RUN;

```

Now that I had lines with the extraneous spaces removed, I wanted to create longer strings. With the 200 character limitation, the simplest thing to do was just put them together two by two.

```

000112 DATA COMP; SET COMP;
000113             BY DN CT;
000114 RETAIN LINE1;
000115 LENGTH NTEXT $200;
000116 IF LN <= 2 THEN NTEXT=TEXT;
000117 ELSE
000118 IF LN > 2 THEN DO;
000119 IF CT IN (1,3,5,7,9,11,13,15,17,19,21,23) THEN DO;
000120     LINE1=TEXT;
000121     IF LAST.DN THEN NTEXT=LINE1;
000122     END;
000123 ELSE IF CT IN (2,4,6,8,10,12,14,16,18,20,22,24) THEN
000124     NTEXT = TRIM(LINE1)||' '||TEXT;
000125     IF NTEXT= ' ' THEN DELETE;
000126     LEN=LENGTH(NTEXT);
000127     END;
000128 RUN;

```

Using PROC REPORT the data is displayed in 70 columns (set by the WIDTH option). The FLOW option wordwraps the text in the column.

```
000129 PROC REPORT;  
000130   COLUMN DN NTEXT;  
000131   DEFINE DN /GROUP NOPRINT;  
000132   DEFINE NTEXT/DISPLAY WIDTH=70 FLOW ' ';  
000133   BREAK AFTER DN /SKIP;  
000134   TITLE ' ';
```

Granted, it's not perfect. The PowerBuilder guys still have to do a bit of editing before they can run one of the SQL chunks because of the word splits, but they are happy with what they have now. Plus, there is the added satisfaction of a "User Programmer" being able to accomplish something fairly easily with SAS which the "real" programmers could not easily do with their tools!

References:

IBM(1995) *Application Programming and SQL Guide Database 2 for MVS/ESA Version 4*

SAS Institute, Inc. (1990), *SAS Language Reference Version 6*, Cary, NC: SAS Institute Inc.

SAS Institute, Inc. (1993), *SAS Technical Report P-258 Using the REPORT Procedure in a Nonwindowing Environment*, Cary, NC: SAS Institute Inc.s

Ray Pass and Daphne Ewing, "So You're Still Not Using PROC REPORT. Why Not?", Proceedings of the Twenty-Third Annual SAS Users Group International Conference (1998), pp. 677-692

Ann Stephan
P O Box 12103
Austin, TX 78711
(512) 463-4430
ann.stephan@cpa.state.tx.us

This is what they initially had to work with:

columns 1 - 72	73 - 80
000001 DBRM μMMLO394 ITS03P ^U} i} B	1 E
000002	
000003 DBRM I DECLARE C1 CURSOR FOR SELECT MAX (APPL_VRSN_TX	) FROM V
000004 VERSION01 WHERE APPL_ID = : H AND SUPT_CD = : H FOR FETCH ONLY	ë
000005 E CHSR1 E CHSR2	
000006 DBRM L aDECLARE C2 CURSOR WITH HOLD FOR SELECT MAX (APP	L_VRSN_T
000007 X) FROM VVERSION01 WHERE APPL_ID = : H AND SUPT_CD = : H FOR FETCH ONLY	
000008) E CHSR1 ? E CHSR2	
000009 DBRM ¼ DECLARE C3 CURSOR FOR SELECT PC_MESSAGE_DEF . ME	SSAGE_GR
000010 OUP , PC_MESSAGE_DEF . MESSAGE_ID , PC_LANGUAGE_DEF . LANGUAGE_ID , PC_M	ESSAGE_D
000011 ETAIL . MESSAGE_TITLE , PC_MESSAGE_DETAIL . MESSAGE_TEXT , PC_MESSAGE_DE	F . SEVE
000012 RITY_LEVEL , PC_MESSAGE_DEF . LOGGING_IND , PC_MESSAGE_DEF . DIALOG_CLAS	SNAME ,
000013 PC_MESSAGE_DEF . RESPONSE_CLASSNAME , PC_MESSAGE_DEF . RESPONSE_DEFAULT	FROM PC_
000014 LANGUAGE_DEF , PC_MESSAGE_DEF , PC_MESSAGE_DETAIL WHERE (PC_MESSAGE_DEF	. MESSA
000015 GE_SKEY = PC_MESSAGE_DETAIL . MESSAGE_SKEY) AND (PC_LANGUAGE_DEF . LAN	GUAGE_SK
000016 EY = PC_MESSAGE_DETAIL . LANGUAGE_SKEY) AND ((PC_LANGUAGE_DEF . LANGU	AGE_ID =
000017 : H)) FOR FETCH ONLY ; A VASR1	

This is what they can get now:

000001 DBRM μMMLO394 ITS03P
000002
000003 DBRM DECLARE C1 CURSOR FOR SELECT MAX(APPL_VRSN_TX) FROM V VERSION01
000004 WHERE APPL_ID = : H AND SUPT_CD = : H FOR FETCH ONLY
000005
000006 DBRM DECLARE C2 CURSOR WITH HOLD FOR SELECT MAX(APPL_VRSN_T X) FROM
000007 VVERSION01 WHERE APPL_ID = : H AND SUPT_CD = : H FOR FETCH ONLY
000008
000009 DBRM DECLARE C3 CURSOR FOR SELECT PC_MESSAGE_DEF.MESSAGE_GR OUP,
000010 PC_MESSAGE_DEF.MESSAGE_ID, PC_LANGUAGE_DEF.LANGUAGE_ID, PC_MESSAGE_D
000011 ETAIL.MESSAGE_TITLE, PC_MESSAGE_DETAIL.MESSAGE_TEXT,
000012 PC_MESSAGE_DEF.SEVE RITY_LEVEL, PC_MESSAGE_DEF.LOGGING_IND,
000013 PC_MESSAGE_DEF.DIALOG_CLASSNAME,
000014 PC_MESSAGE_DEF.RESPONSE_CLASSNAME, PC_MESSAGE_DEF.RESPONSE_DEFAULT
000015 FROM PC_ LANGUAGE_DEF, PC_MESSAGE_DEF, PC_MESSAGE_DETAIL WHERE(
000016 PC_MESSAGE_DEF.MESSA
000017 GE_SKEY = PC_MESSAGE_DETAIL.MESSAGE_SKEY) AND(
000018 PC_LANGUAGE_DEF.LANGUAGE_SK EY = PC_MESSAGE_DETAIL.LANGUAGE_SKEY)
000019 AND((PC_LANGUAGE_DEF.LANGUAGE_ID =
000020 : H)) FOR FETCH ONLY